To Do the communication over the internet I am going to use javas Sockets libery. This means I can have 1 sockets conecction between the server(car) and the client(Phone). From there I have a few options, I can: use a printwriter and buffered reader to send string from the client to server and visa versa. Or I can use an object input/output stream this allows me to send any data type between the server and client. However this could cause errors if the wrong data type is received and a conversion error (EG string to integer) occurs. However if I sent just string then I could convert the string to for example an integer and if there is an error it will be easier to see. If I did it this way then each string sent would lightly contain lots of info ( EG [stateof acceleration]/[stateofturning]/[otherdata]) this could mean that data gets mixed up and cause errors. Further more I belive I can combine data types together in a bundle and send that, this would be the prefured way of doing it, as it will be quick and the data will not get mixed up. When I have internet I will research this more. Also I will use a ^ to distinguish when one data ends and another begins if I cant get the package to work.

UPDATE[date]:

With that in mide the following thing wil be needed to be transfured:

From client to server: Turning state, this will be left right or straight. This could be stored as a integer(0 for left, 1 for straight, 2 for right) or a string(straight, left, right) but storing it a integer is smaller and more effective. I will also need acceleration, this could be stored as a integer(0=folward, 1 = stationery, 2= backwards) or as a string but as I have discussed that is ineffitent, or as a integer where you have varying levels of acceleration, for this I would need a car that supports it whitch most do not and it would add extra complexity. Then a string for extra commands is needed (EG: Discconect or error). I will need to do more resuch in to if any more data is needed.

From Server to Client: First a veriable stating that no error acored, and if it did then they disconnect from eachother and send the error, if there is no error then the word good is sent. I don’t believe there is anything else I need but I will run in to that and update this if there is.

Conclution:

Client to server: Command(string) then Acceleration(int) then Turning(int).

I will use ^ to separate data

Server to Client: Error(string)

**Extra**

I have decided to use the port 40506 because it appears to be unused by any other application.

Because I am programing this in java I want to do it in a object oriented manner. This is good because it means the code will be easily transferable to other projects and it will be easy to debug.

29-7-14: I believe that sending a string containing the command to be executed from the client to the server is inefficient. So instead I will use an int will a number mapped to each command.

Commands:

0: Non

1: Disconnect

More can be added as needed but this is the only one I can think of at the moment.

It is also inefficient that I will be sending each int individually. So I could do a few things: use an array of length 3 and place the command at position 0, acceleration at 1 and turning at 2. Or I could use a bundle. I have desided to use a array because a bundle is desighned to be sent from class to class not over the internet, as a result they can be complex and large in size. Whereas an array can be sent using an Objectstream[clarification] and is not bigger than it needs to be.

To Conclude: I will use an array of length 3, position 0 is the command (int), position 1 is the acceleration value(int) and 2 is the turning value(int).

31-7-14: I will use an update rate of 16, this means that 16 times per second the app will tell the sever what buttons are pressed. This means that the app will send data every 1/16seconds

16-8-14: I Will note send data from the server to the client because if there was an error then one would be thrown and I have the nessasery routines to deal with that and there is nothing that I have found that needs to be passed to the client.

25-8-14: I changed it so I am now using UDP as explained in the RCserver doc.

26-8-14: because I am using UDP I am now transfuring 3 4 byte/32 bit strings containing an int. also because of the new server desighn and the next part I don’t need a command because there is not active conection to close and I don’t want to stop the whole app on the pi to stop. Because UDP packets are not nesaseraly recived in the order they are sent I am making it so that they are not confused(front and side) so front will transfer a value of 0, 1 or 2 and side will transfer a value of 3, 4 or 5.

27-8-14: I will be updating the server 32 times a second now as now I am using UDP I believe it can support that without putting strain on the network.